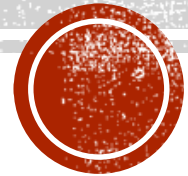


# PRINCIPLES OF DATA ORGANISATION

Larson & Kajla



# MOTIVATION

- Key, pointer pairs ~ index
- Hashed file organization
- External static hashing
  - We need to know the size of the data beforehand



# LARSON & KAJLA

- Perfect hashing introduced in 1984
- Uses two sets of hash functions
  - $h_i(k), i \in \{1, \dots, M\}$  generates a sequence of **page addresses** where a record with a key  $k$  could be inserted
  - $s_i(k), i \in \{1, \dots, M\}$  generates a sequence of  $d$ -bit long strings called **signatures**
- Pages have assigned  $d$ -bit long strings called **separators**
  - Restrict the values that can be inside a page (“height of the door”)
  - A record with a key  $k$  can be inserted into page determined by  $h_i(k)$  (or can be found in) only if its signature  $s_i(k)$  is smaller than the separator of that page



# LARSON & KAJLA

- ↳ Records are **sorted** in the page according to increasing values of their signatures
  - ↳ The approach is optimized for reading (static hashing)
- ↳ Page separator is the lowest signature of all the records which could not fit into that page (overflowed records)
- ↳ The initial value of the separator is  $2^{d-1}$ 
  - ↳  $d$  = length (in bits) of a signature
  - ↳ Signatures cannot take this value
- ↳ During the INSERT operation, more records can be pushed out of the page → **INSERT cascade**



# LARSON & KAJLA – EXAMPLE

Insert a record with key 'ab'

$$\begin{aligned}
 h_1(ab) &= 10 & h_2(ab) &= 46 \\
 s_1(ab) &= 1011 & s_2(ab) &= 0101 \\
 b &= 3
 \end{aligned}$$

10	46	95	116
od-0100	ef-0100	kl-0100	op-0010
	gh-1000	mn-1001	
	ij-1000		
sep: 1000	sep: 1001	sep: 1111	sep: 1000

The target page for 'ab' is 46, so it pushes out records with keys 'gh' and 'ij' ~ new page separator.

$$\begin{aligned}
 h_j(gh) &= 46 & h_{j+1}(gh) &= 95 \\
 s_j(gh) &= 1000 & s_{j+1}(gh) &= 1011
 \end{aligned}$$

$$\begin{aligned}
 h_j(ij) &= 46 & h_{j+1}(ij) &= 116 \\
 s_j(ij) &= 1000 & s_{j+1}(ij) &= 0100
 \end{aligned}$$

10	46	95	116
od-0100	ef-0100	kl-0100	op-0010
	ab-0101	mn-1001	ij-0100
		gh-1011	
sep: 1000	sep: 1000	sep: 1111	sep: 1000



# LARSON & KAJLA – ALGORITHM

```
void ACCESS(int sep[], KEY_TYPE k, PAGE_TYPE &page, bool &found) {  
    int m = sep::size();  
    for (int i = 0; i < m; i++) {  
        int adr = hi(k);  
        sign = si(k);  
        if (sign < sep[adr]) {  
            GET_PAGE(adr, page);  
            found = SEARCH_PAGE(page, k);  
            return;  
        }  
    }  
    found = FALSE;  
}
```

Load from  
secondary  
memory



# LARSON & KAJLA

## Hash function

$$h_0(k) = k \bmod M$$

$$h_{i+1}(k) = (h_i(k) + \text{step}) \bmod M, \text{step} = (\lfloor k/M \rfloor \bmod (M-2)) + 1$$

$M$  should be prime number so that all pages can be visited

## Signature function

$$s_i(k) = (r_i(k) \bmod e) \bmod 2^d$$

$r_i$  – generates a random number

$e, d$  – constants

Knuth's random number generator works with binary string of the key  $k'$  (if  $k$  is not a number it needs first be converted to it)

$$r_0(k') = k'$$

$$r_{i+1}(k') = (a * r_i(k') + c) \bmod 2^{32}, a = 3141592653, c = 2718281829$$

Larson used  $e = 2^{13} - 1 = 8191$  to get suitable numbers from  $r$   
 $2^d$  secures that the signature will contain  $d$  bits

